

Generalized Linear Models in R

Andrew McAdam

2020-11-22

Contents

Generalized Linear Models	1
Link Functions	1
Why not just transform the data and do a glm?	2
How does GLiM estimate the paramters?	2
Implementing a GLiM in R	2
Meta-Data	3
Specifying a GLiM in R	3
Testing Assumptions in GLiMs	5
Diagnostic Plots	5
Over-Dispersion	11
Nonlinearities	15
Probit Model	23
Interpreting parameters in a GLiM	25
Residuals	27

Generalized Linear Models

We have spent a fair amount of time going over general linear models, but these are just special cases of general-IZED linear models for which the errors are normally distributed. But in many cases our observations are bounded (i.e. must be positive, or must be between zero and one), which can lead to errors that are not normally distributed. Luckily, generalized linear models (GLiMs) are very flexible and can handle any error distribution from an exponential family of distributions (e.g. normal, Poisson, binomial, multinomial, exponential, gamma, negative binomial). Some of these might be familiar to you. One of the most ‘famous’ GLiMs is a logistic regression in which we are trying to predict a response variable that can take on values of zero or one (or a proportion) by a set of continuous or categorical covariates.

Link Functions

One key feature of a GLiM is that it involves a *link function*, which is a function that is applied to the response variable (like a transformation). For example, in a Poisson GLiM the default link function is a log link function. As a result we are modelling the $\log(Y)$ rather than Y . This is important to remember because if we are to make sense of the parameters in the model we need to remember that the parameters will predict $\log(Y)$ (i.e. on the link scale) rather than Y (i.e. on the raw scale). It is also worth pointing out that you

don't need to worry about adding a small constant to your response variable if there are zeros (because $\log(0)$ is undefined) like you would if you were transforming your response variable - the link function takes care of this. Similarly, the default link function for a binomial model is the logit link function.

In addition to a link function, GLiMs also specify a variance function. The variance function describes the relationship between the mean and the variance. This allows for proper calculation of the variance under non-normal conditions. This ability to separately specify the link function and the variance function in a GLiM provides a lot of flexibility to be able to put together an appropriate model.

Why not just transform the data and do a glm?

Transformations often do a pretty good job of stabilizing the variance, but they often have trouble fixing the skewness in the data. We are also often wanting to both linearize a relationship and normalize the residuals with a single transformation and this is often a tall task. So in most cases, a GLiM is preferable to transforming the response variable to run a glm.

How does GLiM estimate the parameters?

Whereas a glm solves for the set of parameters by minimizing the squared residuals (sums of squares approach), this is not possible for GLiMs. Instead they use an iterative procedure called *iterative weighted least squares*. This cannot be done by hand, but is easy to do with a computer. This is why logistic regression only became a popular statistical approach in biology in the 1970s.

Implementing a GLiM in R

In order to run a GLiM in R, we need to specify the link function and the distribution of the errors. Otherwise the model specification is the same.

There are some additional packages that we will load for GLiMs

```
library (gam)
```

```
## Loading required package: splines
```

```
## Loading required package: foreach
```

```
## Loaded gam 1.20
```

```
library (MASS)
```

```
library (boot)
```

We are going to return to the lizard egg data to start with. I am going to load a new data file.

```
egg2<-read.table("data/egg.temp2.csv", sep="," , header=T)
```

```
summary (egg2)
```

```
##      LayDay      clutch      bin      postlay
## Min.   : 75.0   Min.   :1.000   Min.   : 19.0   Min.   :3.250
## 1st Qu.: 96.0   1st Qu.:1.000   1st Qu.:115.0   1st Qu.:4.430
## Median :115.0   Median :2.000   Median :136.0   Median :4.810
## Mean   :118.6   Mean   :1.653   Mean   :133.5   Mean   :4.832
## 3rd Qu.:140.0   3rd Qu.:2.000   3rd Qu.:158.0   3rd Qu.:5.210
## Max.   :188.0   Max.   :4.000   Max.   :178.0   Max.   :7.220
## clutch.size  eggcode      eggmass  post.eggmass
## Min.   :1.000  Length:1247      Min.   :0.26   Min.   :0.1900
```

```
## 1st Qu.:4.000   Class :character   1st Qu.:0.40   1st Qu.:0.3500
## Median :5.000   Mode  :character   Median :0.44   Median :0.4100
## Mean   :5.018                   Mean   :0.44   Mean   :0.4048
## 3rd Qu.:6.000                   3rd Qu.:0.48   3rd Qu.:0.4500
## Max.   :9.000                   Max.    :0.91   Max.    :0.9100
##   dam.oby           hatch
## Length:1247         Min.    :0.0000
## Class :character   1st Qu.:0.0000
## Mode  :character   Median  :1.0000
##                               Mean    :0.7298
##                               3rd Qu.:1.0000
##                               Max.    :1.0000
```

Meta-Data

These data were from a captive breeding experiment I ran with side-blotched lizards (*Uta stansburiana*) while I was a postdoc at UC Santa Cruz with Barry Sinervo. Multiple female lizards and one male were housed within a single holding area (bin).

You should get used to writing out similar data descriptions in your own Rmd files. They are very helpful!

- *LayDay* - day of the year (Julian date) on which the egg was laid.
- *clutch* - whether the clutch was the first, second, third, or fourth clutch of the year.
- *bin* - the housing bin in which the female was raised. There were more than one female per bin.
- *postlay* - mass of female after she finished laying her eggs (g?).
- *clutch.size* - number of eggs in the clutch.
- *eggcode* - whether or not the egg was miniaturized by removing some yolk. (C = control; M = miniaturized).
- *eggmass* - mass of the egg (g) when laid.
- *post.eggmass* - mass of the egg (g) after yolk was removed.
- *dam.oby* - throat colour genotype of the mother.
- *hatch* - whether or not the egg successfully hatched (0 = no; 1 = yes).

This datafile is similar to the `egg.temp.csv` file that we have worked with in the past, but this one also includes a response variable that indicates whether or not the egg hatched (`hatch`)

Note that about 73% of the eggs hatched We will need to change “bin” to a factor

```
egg2$bin<-factor(egg2$bin)
```

Specifying a GLiM in R

We will start with a GLiM to predict whether or not an egg will hatch. This is a binary response so we will be dealing with a binomial family. In this case the GLiM is the equivalent of a logistic regression.

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, family=binomial, data=egg2)
```

Note that we are using `glm` instead of `lm` and we therefore need to specify the family type. We did not specify a link function so the default “link=logit” is used.

```
summary (hatch.glm)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial,
##      data = egg2)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
```

Note that this is very similar to what we are used to seeing. Instead of dealing with variance though we are dealing now with DEVIANCE. Deviance is defined to be twice the difference between the log-likelihood of the best possible model and the log-likelihood of the current model. The log-likelihood of the best possible model is based on the variance in your response variable.

Also note that we are no longer testing the significance of our parameters based on a t test statistic. Instead we are now using a Z test statistic. This test statistic does not have an associated degrees of freedom. So when you report the significance of a parameter in a GLiM you would report b+/-SE, Z and P (no df needed).

The proportion of deviance explained is

```
1-1375.4/1455.3
```

```
## [1] 0.05490277
```

This is the summary command but we can also test the significance of terms rather than parameters using the anova command.

```
anova (hatch.glm, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
```

```
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                1246      1455.3
## eggcode             1   32.635      1245      1422.6 1.112e-08 ***
## LayDay              1   33.547      1244      1389.1 6.955e-09 ***
## post.eggmass       1   13.635      1243      1375.5 0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that in this case the terms are assessed from first to last. The model is assessed in a “Type I” style where the order matters. This analysis is basically a series of likelihood ratio tests, comparing progressively more complex models.

Testing Assumptions in GLiMs

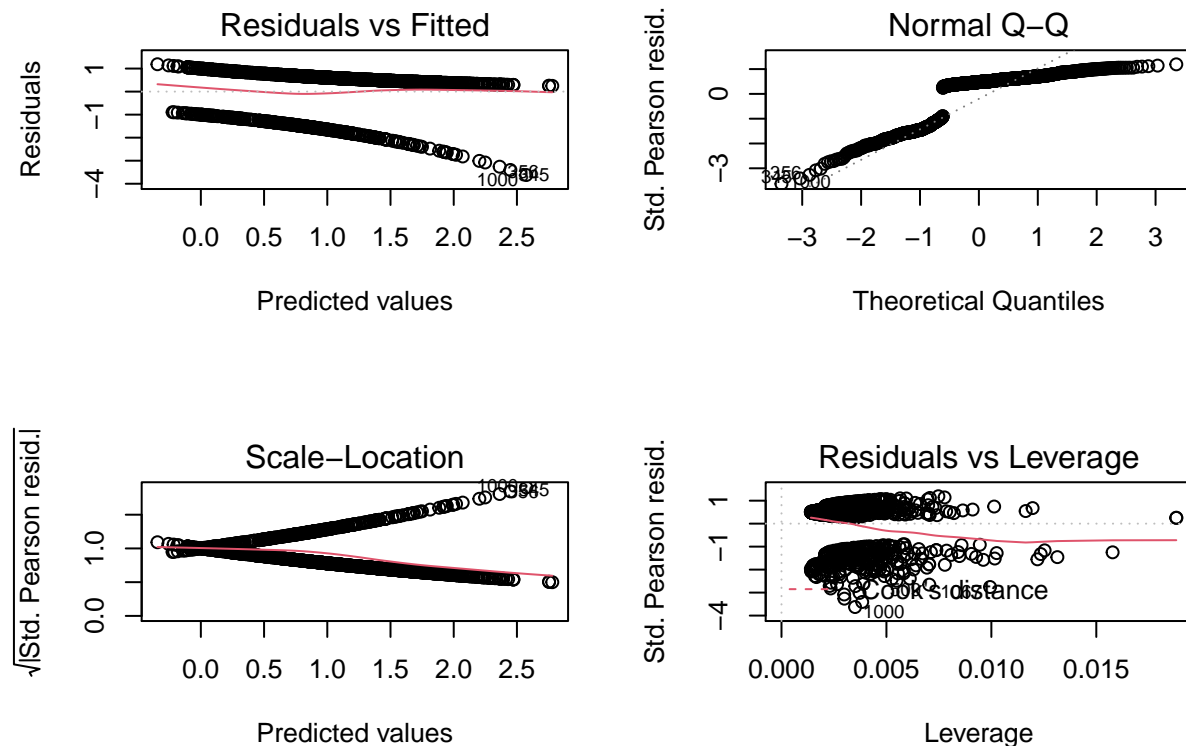
Many of the assumptions of a GLiM are similar to the assumptions of a glm. These include:

- Independence of observations - good sampling
- Correct specification of the variance function
- Correct specification of the link function
- Correct form of the explanatory variables - look for nonlinearities
- Lack of undue influence of individual observations – Cook’s distance
- Correct specification of the dispersion parameter

Diagnostic Plots

Regular diagnostic plots don’t help us very much for GLiMs.

```
par (mfrow=c(2,2))
plot (hatch.glm)
```



Note that the residuals are never going to look very good in a binomial glm. We are used to using these plots to test for homogeneity in the residuals, but in a GLiM the residuals are never going to appear homogeneous, because the response variable is often very discontinuous. Nevertheless, we can still use these diagnostics to look at points with high leverage and high residuals (the last plot - Cook's distances).

There is a nice package for assessing diagnostic plots for GLiMs. More information can be found at: <https://theoreticalecology.wordpress.com/2016/08/28/dharma-an-r-package-for-residual-diagnostics-of-glmms/>

Here is the Vignette: <https://cran.r-project.org/web/packages/DHARMA/vignettes/DHARMA.html>

I will use it here.

```
#install.packages("DHARMA")
library(DHARMA)
```

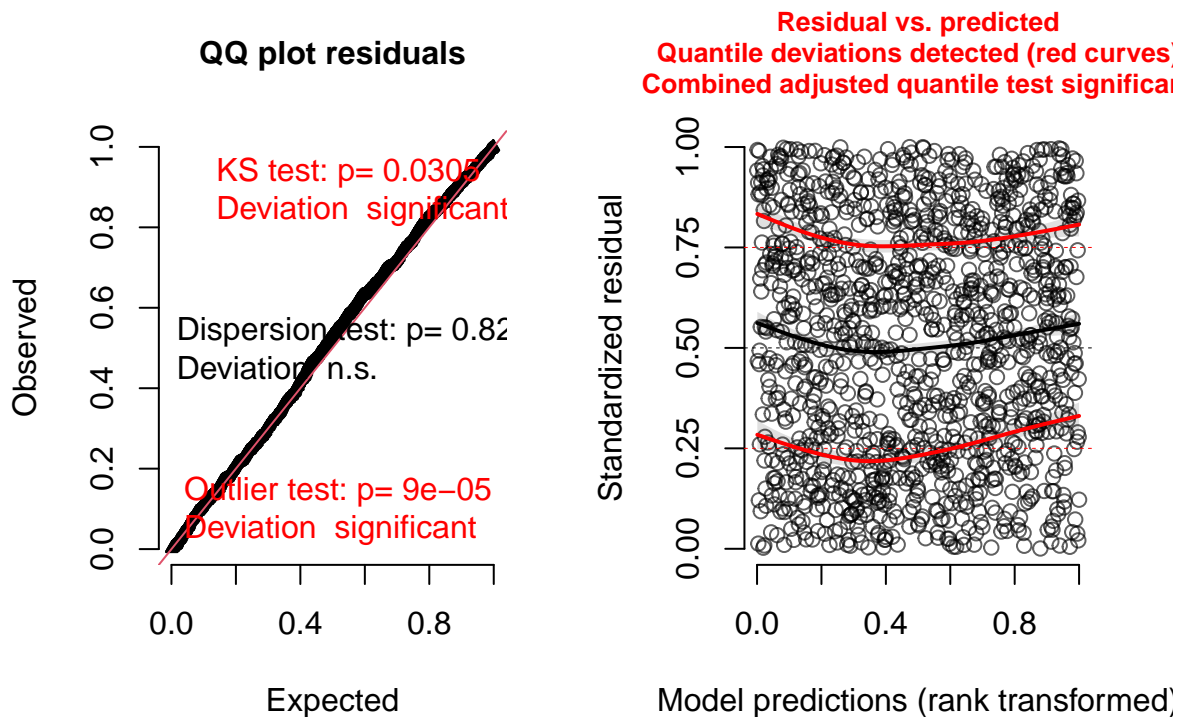
```
## This is DHARMA 0.3.2.0. For overview type '?DHARMA'. For recent changes, type news(package = 'DHARMA')
simulation_output_hatch.glm<-simulateResiduals(fittedModel = hatch.glm, n = 250)
```

Here we now expect a uniform distribution (flat distribution) of simulated residuals.

We can visualize these simulated residuals using

```
plot(simulation_output_hatch.glm)
```

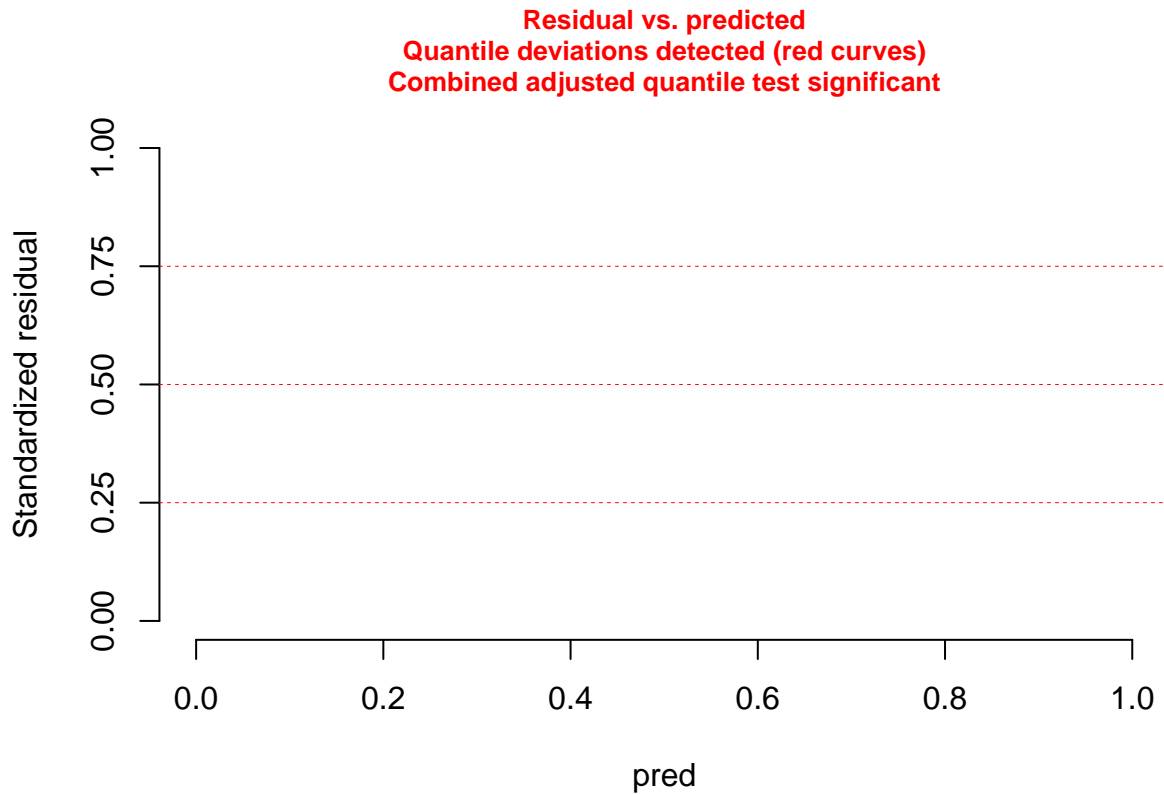
DHARMA residual diagnostics



So the above plot shows no deviation from the expected distribution. In the plot on the right, 'outliers' are plotted as red stars. The red lines in the plot on the right should be straight across and at the values on the y axis of 0.25, 0.5 and 0.75. These look very good above!!

We should also test whether there is a relationship between the residuals and our specific predictors

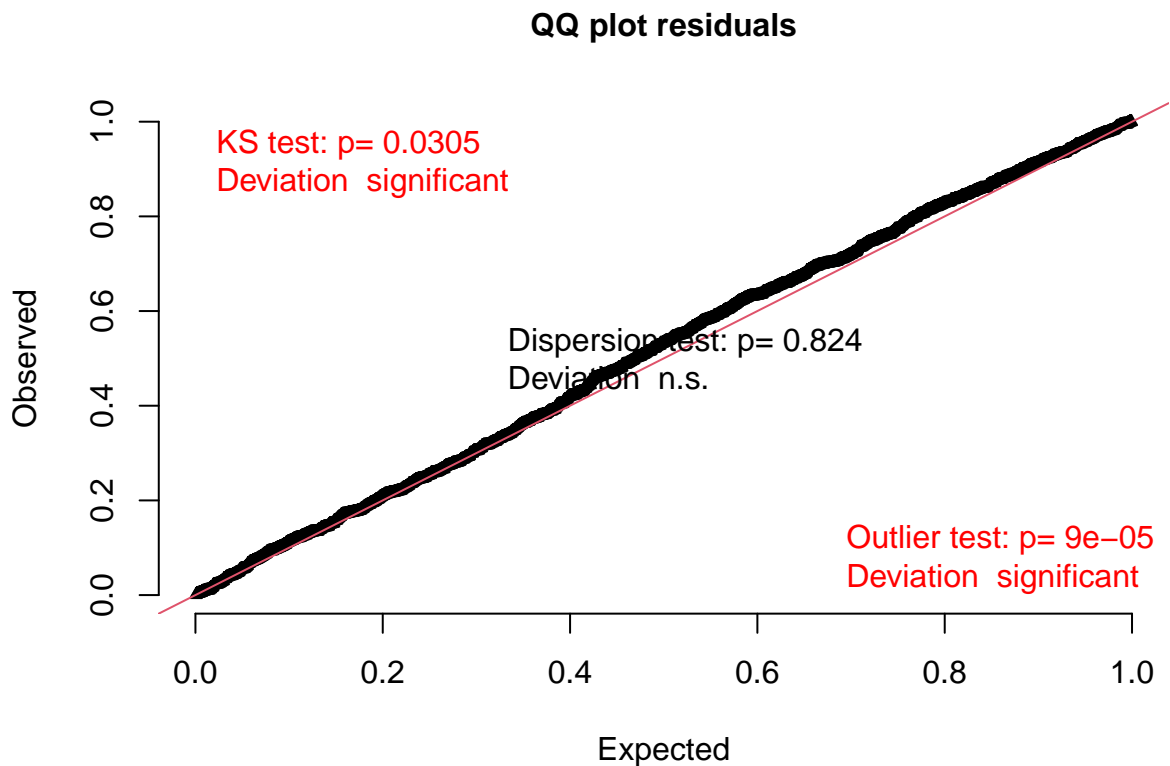
```
plotResiduals(egg2$LayDay, simulation_output_hatch.glm$scaledResiduals)
```



This also looks very good.

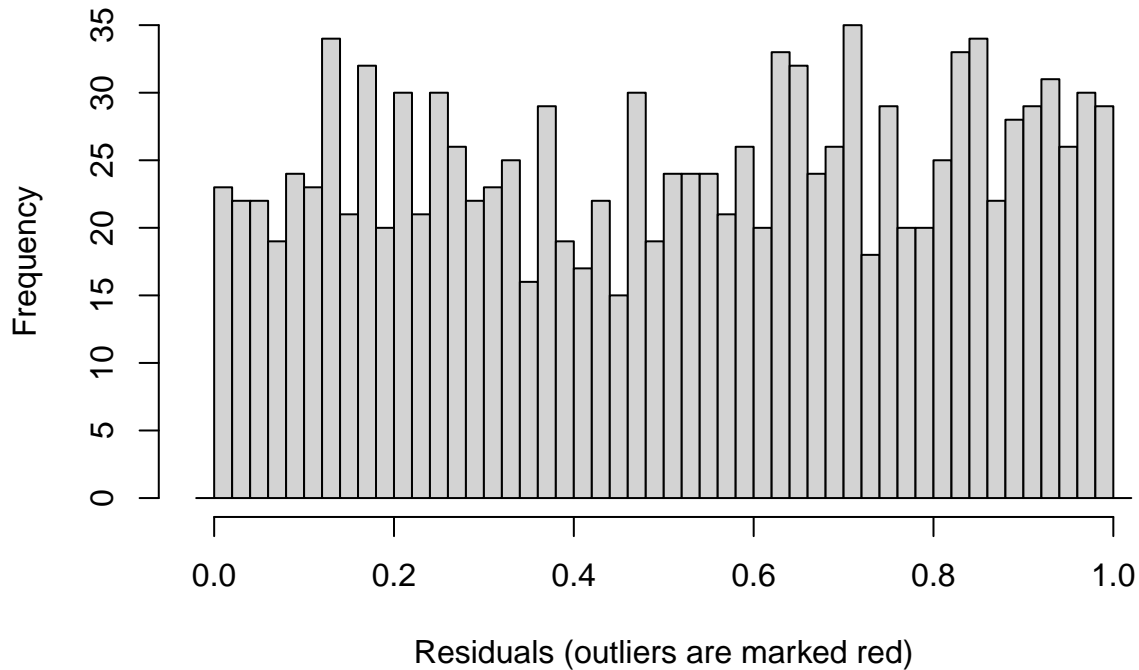
This package also runs some specific tests, which seem very nice!

```
testUniformity(simulation_output_hatch.glm)
```



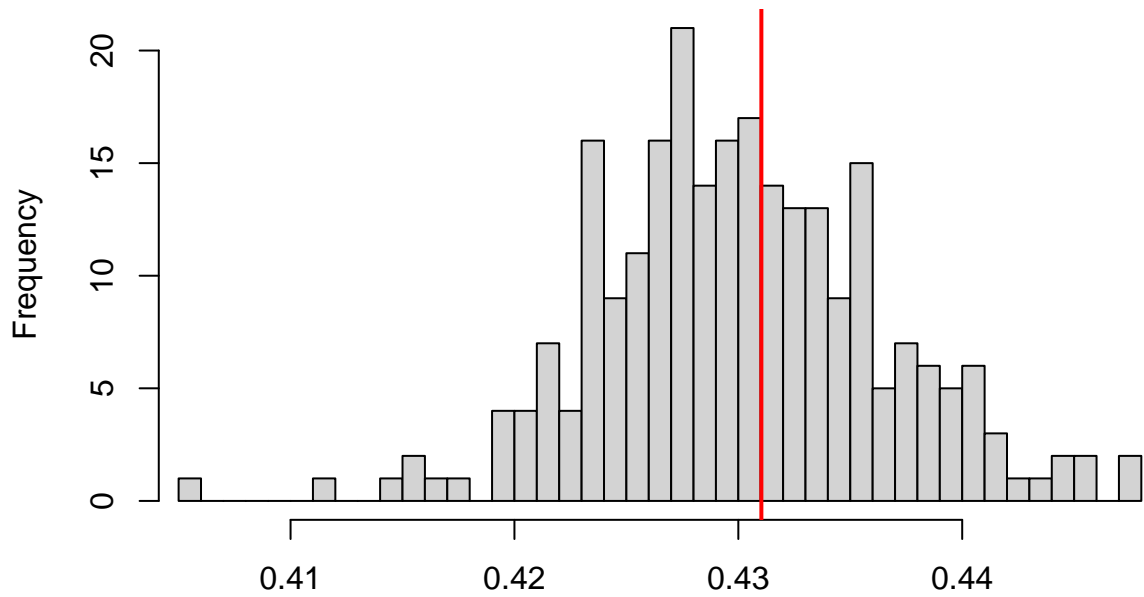
```
##
## One-sample Kolmogorov-Smirnov test
##
## data: simulationOutput$scaledResiduals
## D = 0.040955, p-value = 0.0305
## alternative hypothesis: two-sided
testOutliers(simulation_output_hatch.glm)
```

Outlier test significant



```
##
## DHARMA outlier test based on exact binomial test
##
## data: simulation_output_hatch.glm
## outliers at both margin(s) = 0, simulations = 1247, p-value = 8.536e-05
## alternative hypothesis: true probability of success is not equal to 0.007968127
## 95 percent confidence interval:
## 0.000000000 0.002953832
## sample estimates:
## frequency of outliers (expected: 0.00796812749003984 )
## 0
testDispersion(simulation_output_hatch.glm)
```


DHARMA nonparametric dispersion test via sd of residuals fitted vs. simulated

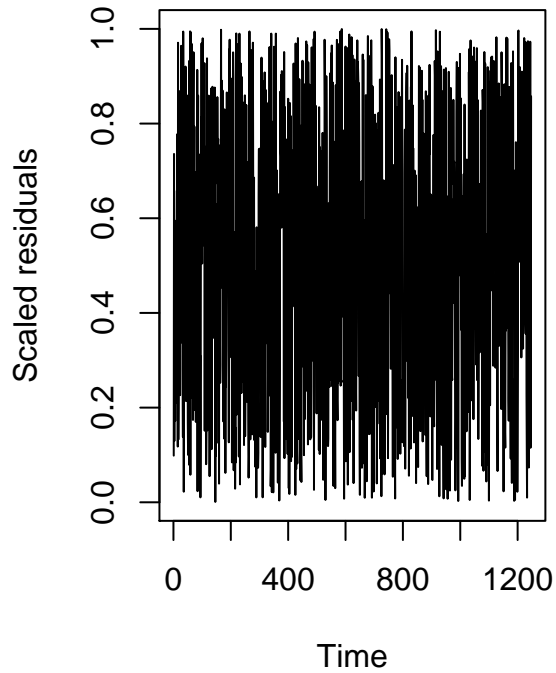


Simulated values, red line = fitted model. p-value (two.sided) = 0.824

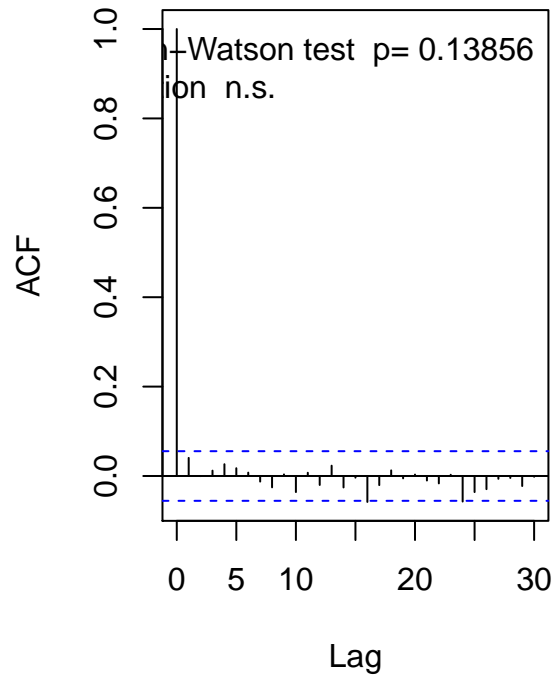
```
##
## DHARMA nonparametric dispersion test via sd of residuals fitted vs.
## simulated
##
## data: simulationOutput
## ratioObsSim = 1.0023, p-value = 0.824
## alternative hypothesis: two.sided
#testZeroInflation(simulation_output_hatch.glm)
#testGeneric(simulation_output_hatch.glm)
testTemporalAutocorrelation(simulation_output_hatch.glm)

## DHARMA::testTemporalAutocorrelation - no time argument provided, using random times for each data po
```

Residuals vs. time



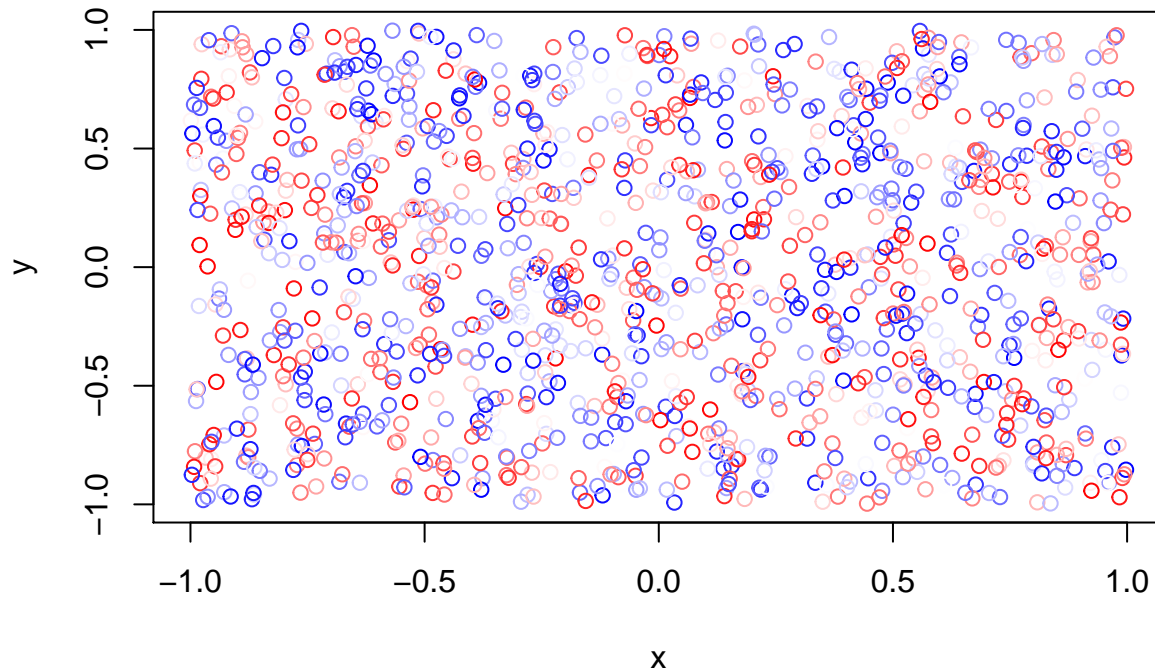
Autocorrelation



```
##
## Durbin-Watson test
##
## data: simulationOutput$scaledResiduals ~ 1
## DW = 1.9162, p-value = 0.1386
## alternative hypothesis: true autocorrelation is not 0
testSpatialAutocorrelation(simulation_output_hatch.glm)

## DHARMA::testSpatialAutocorrelation - no x coordinates provided, using random values for each data po
## DHARMA::testSpatialAutocorrelation - no x coordinates provided, using random values for each data po
```

DHARMa Moran's I test for spatial autocorrelation



```
##
## DHARMa Moran's I test for spatial autocorrelation
##
## data: simulation_output_hatch.glm
## observed = 0.00310943, expected = -0.00080257, sd = 0.00205912, p-value
## = 0.05745
## alternative hypothesis: Spatial autocorrelation
```

Over-Dispersion

One important test above is the test for overdispersion. In a GLiM the error variance is not estimated. Instead it is fixed at some value based on the variance function that is determined by the assumed distribution. If your data don't fit this assumed distribution well then the model could be over-dispersed (too much unexplained variation) or under-dispersed (too little variation). Overdispersion can cause your inferences to be overly liberal (too likely to reject the null; p-values are too small) so it is of concern. There are several possible causes of over-dispersion including:

- unexplained variation - try including an additional predictor to account for this variation. Sometimes, however, there is nothing you can do.
- structure in the data - if there is a lack of independence in the data this can lead to overdispersion. Fitting a random effect (see the section on mixed effect models) can sometimes account for this.
- a mis-specified model - perhaps your assumed distribution is not a good one.

The DHARMa package provides an example of what overdispersion would look like.

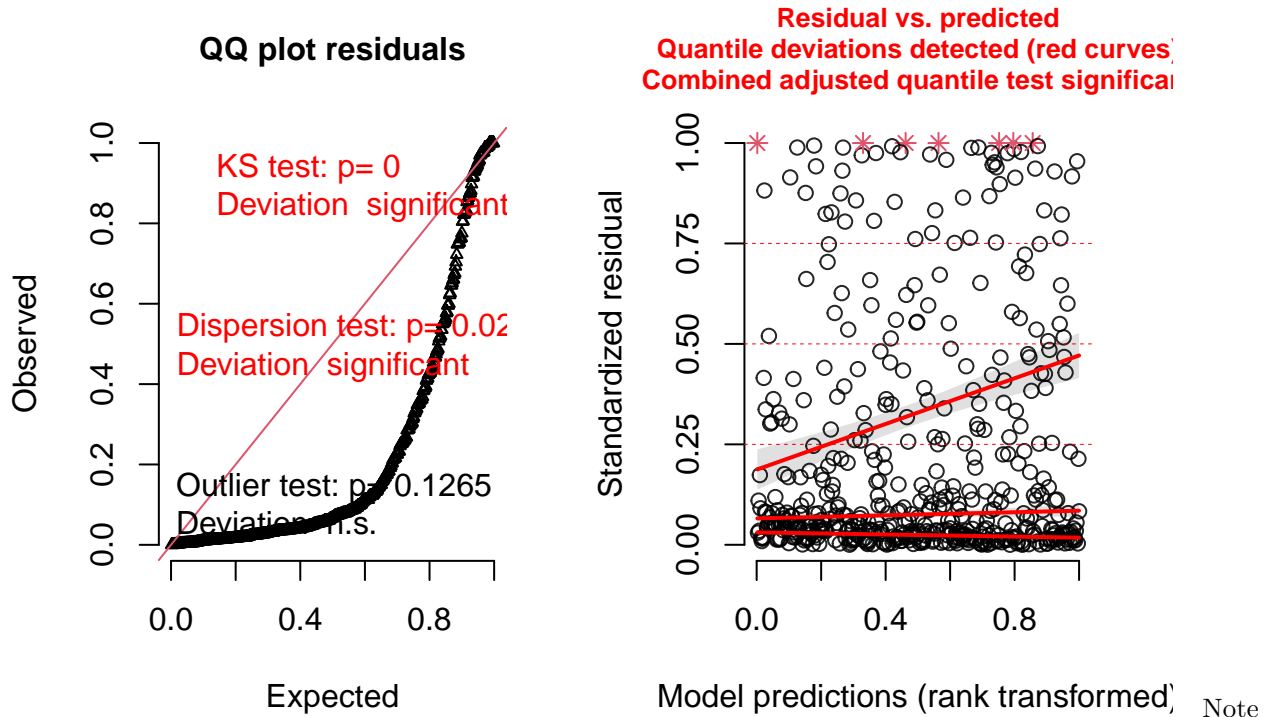
```
library(lme4)
```

```
## Loading required package: Matrix
```

```
testData = createData(sampleSize = 500, overdispersion = 2, family = poisson())
fittedModel <- glmer(observedResponse ~ Environment1 + (1|group) , family = "poisson", data = testData)

simulationOutput <- simulateResiduals(fittedModel = fittedModel)
plot(simulationOutput)
```

DHARMA residual diagnostics



that instead of a uniform distribution of simulated residuals they are clustered around zero.

Here is an example of under-dispersion where we have too few residuals around the tail - too many around the value of 0.5

```
testData = createData(sampleSize = 500, intercept=0, fixedEffects = 2, overdispersion = 0, family = poisson())
fittedModel <- glmer(observedResponse ~ Environment1 + (1|group) , family = "poisson", data = testData)
```

```
## boundary (singular) fit: see ?isSingular
```

```
summary(fittedModel)
```

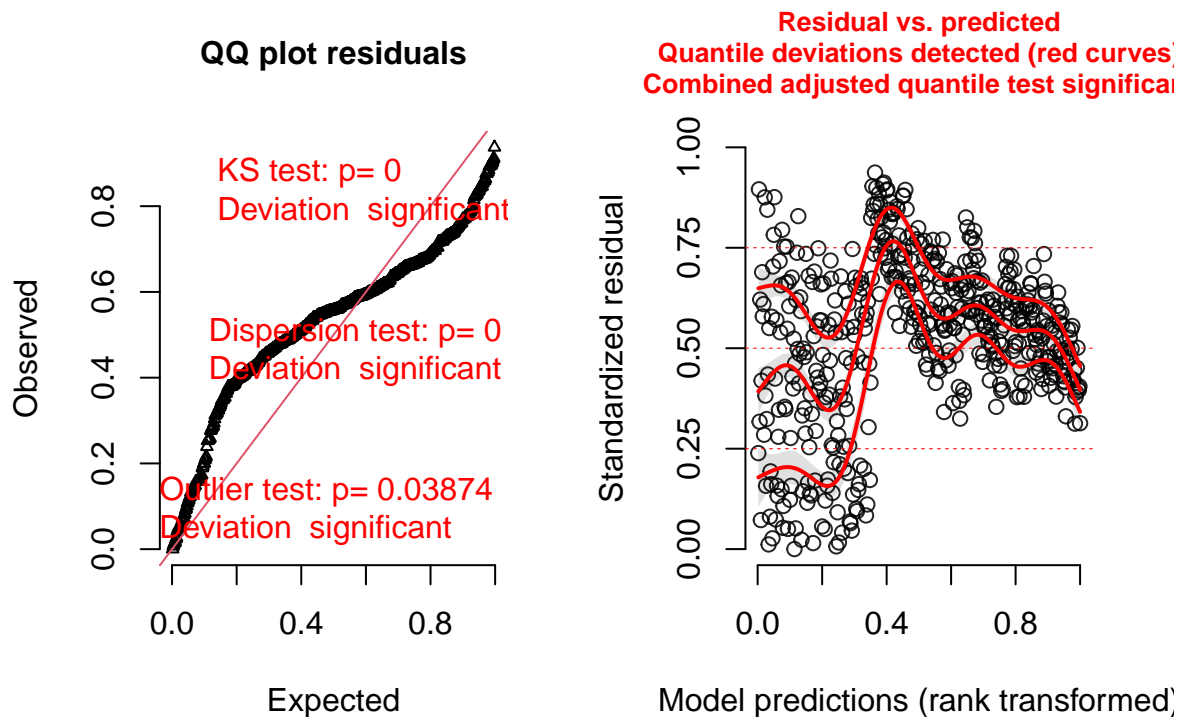
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: poisson ( log )
## Formula: observedResponse ~ Environment1 + (1 | group)
## Data: testData
##
##      AIC      BIC    logLik deviance df.resid
##  963.2   975.9   -478.6   957.2     497
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -0.6051 -0.3621 -0.1020  0.2040  1.0354
##
```

```

## Random effects:
## Groups Name      Variance Std.Dev.
## group (Intercept) 0          0
## Number of obs: 500, groups: group, 10
##
## Fixed effects:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.20609   0.06064  -3.399 0.000677 ***
## Environment1  2.29772    0.08628  26.632 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr)
## Environmnt1 -0.819
## convergence code: 0
## boundary (singular) fit: see ?isSingular
simulationOutput <- simulateResiduals(fittedModel = fittedModel)
plot(simulationOutput)

```

DHARMA residual diagnostics



As far as I can tell this DHARMA package is great for testing these assumptions.

There are, however, some simpler ways to test for over-dispersion. As a start, we can estimate the amount of dispersion in a model by comparing the residual df to the residual deviance in a GLiM. We expect that these ought to be similar. NOTE that this is just an estimate.

```
summary (hatch.glm)
```

```
##
## Call:
```

```
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial,
##     data = egg2)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
```

Note the similarity between the residual deviance and the residual df.

When you are looking at the relationship between the residual deviance and the residual df that this is only a reliable measure of overdispersion if $n \cdot p$ (for binomial) is large

```
length(egg2$hatch)*mean(egg2$hatch)
```

```
## [1] 910
```

This is large

We could still get a measure of the dispersion parameter by using the quasi family

```
hatch.glm4<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=quasibinomial, data=egg2, na.action=na.omit)
summary(hatch.glm4)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##     family = quasibinomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -9.982e+00  1.513e+00  -6.597 6.18e-11 ***
## eggcodeM     -5.332e-01  1.524e-01  -3.498 0.000485 ***
## LayDay        1.528e-01  2.452e-02   6.231 6.33e-10 ***
## I(LayDay^2)  -5.717e-04  9.882e-05  -5.785 9.15e-09 ***
## post.eggmass  3.945e+00  1.096e+00   3.600 0.000331 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.9952764)
##
## Null deviance: 1455.3 on 1246 degrees of freedom
## Residual deviance: 1343.0 on 1242 degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

Note that now the dispersion parameter is estimated rather than assuming that it is 1. Notice that the estimated parameter is very close to one.

What would a very poorly fit model look like (i.e. the wrong link)?

```
hatch.poor<-glm(post.eggmass~eggcode+I(LayDay^2), family=quasibinomial, data=egg2, na.action=na.omit)
```

In this case the response variable is normally distributed but we have specified the quasibinomial family

```
summary (hatch.poor)
```

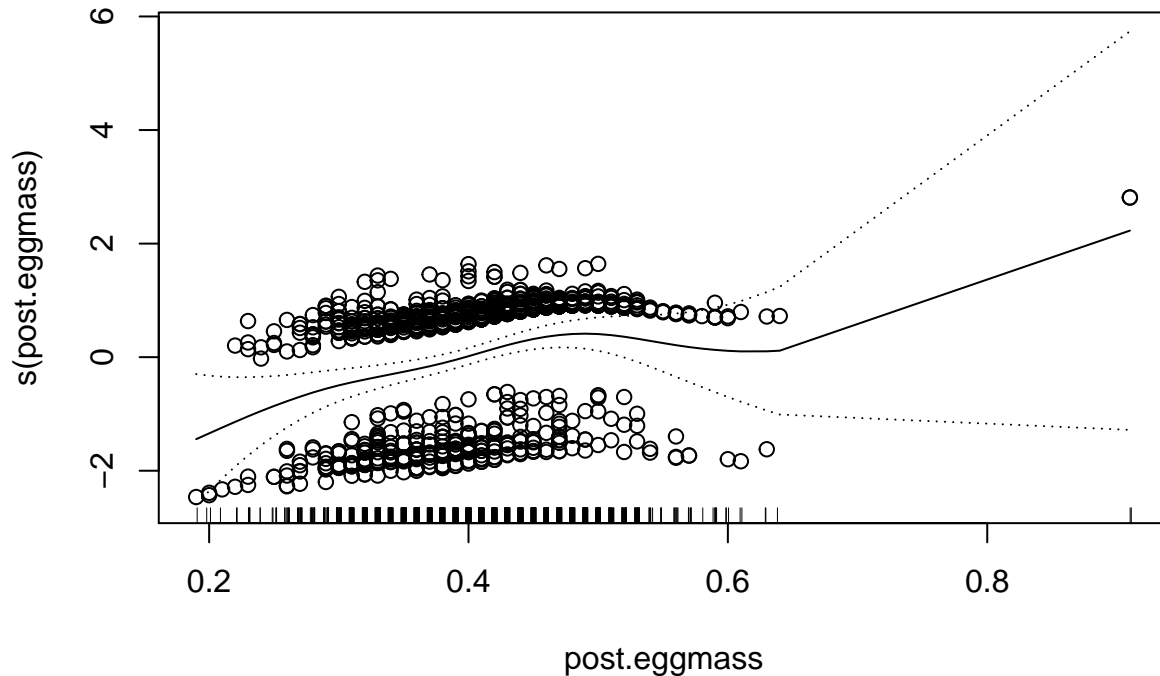
```
##
## Call:
## glm(formula = post.eggmass ~ eggcode + I(LayDay^2), family = quasibinomial,
## data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -0.42469  -0.08084  -0.00374   0.07454   1.05018
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.102e-01  1.952e-02 -21.015  <2e-16 ***
## eggcodeM    -2.967e-01  1.538e-02 -19.292  <2e-16 ***
## I(LayDay^2)  9.693e-06  1.163e-06   8.333  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasibinomial family taken to be 0.01680608)
##
## Null deviance: 28.533 on 1246 degrees of freedom
## Residual deviance: 21.242 on 1244 degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 3
```

Notice that the model is highly under-dispersed. This is evident from the residual deviance and df as well as the estimated dispersion parameter.

Nonlinearities

We should also look for nonlinearities

```
hatch.gam<-gam(hatch~eggcode+s(LayDay)+s(post.eggmass), family=binomial, data=egg2, na.action=na.omit)
anova (hatch.gam)
```

Note that we seem to have some sort of quadratic effect of LayDate and maybe an outlier for post.eggmass. I would normally go back and double-check that value but I remember that egg. It is not a mistake. We might want to try and run the model without this point just to make sure that our conclusions don't depend on that data point. The Cook's distance values suggest that the results do not, but we could check.

So let's fit a quadratic effect for LayDay

```
hatch.glm2<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=binomial, data=egg2, na.action=na.omit)
summary(hatch.glm2)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##      family = binomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.982e+00  1.517e+00  -6.582 4.65e-11 ***
## eggcodeM     -5.332e-01  1.528e-01  -3.490 0.000483 ***
## LayDay        1.528e-01  2.458e-02   6.216 5.09e-10 ***
## I(LayDay^2)  -5.717e-04  9.905e-05  -5.772 7.85e-09 ***
## post.eggmass  3.945e+00  1.098e+00   3.591 0.000329 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1343.0  on 1242  degrees of freedom
```

```
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4

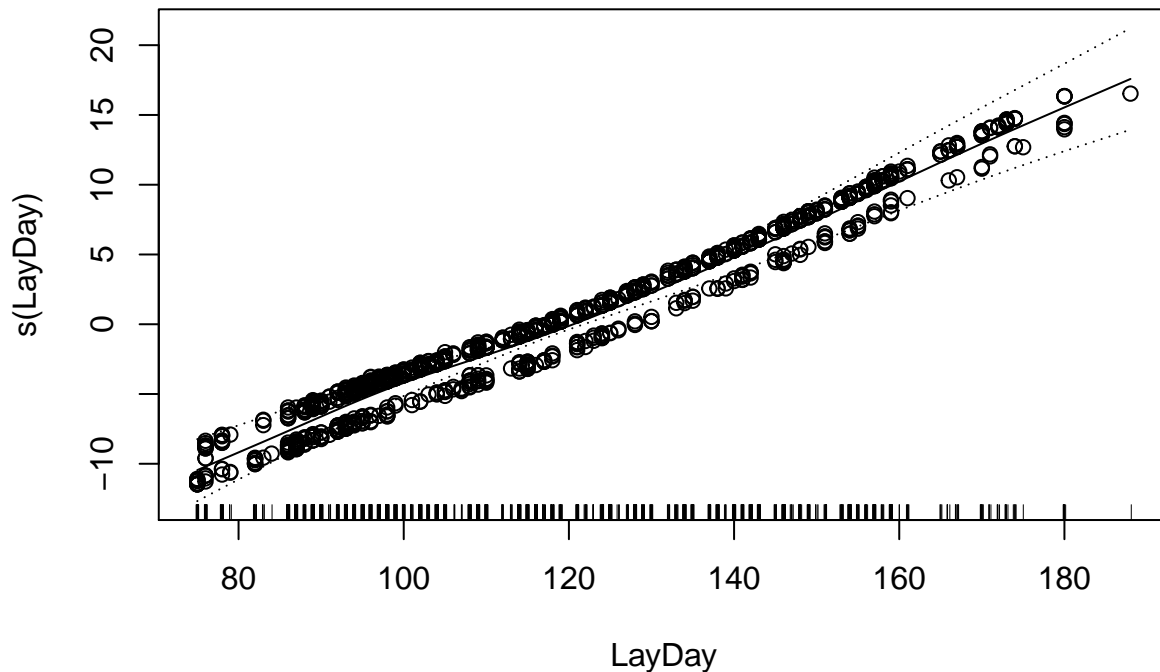
Do any non-linearities remain after we consider the quadratic effect?

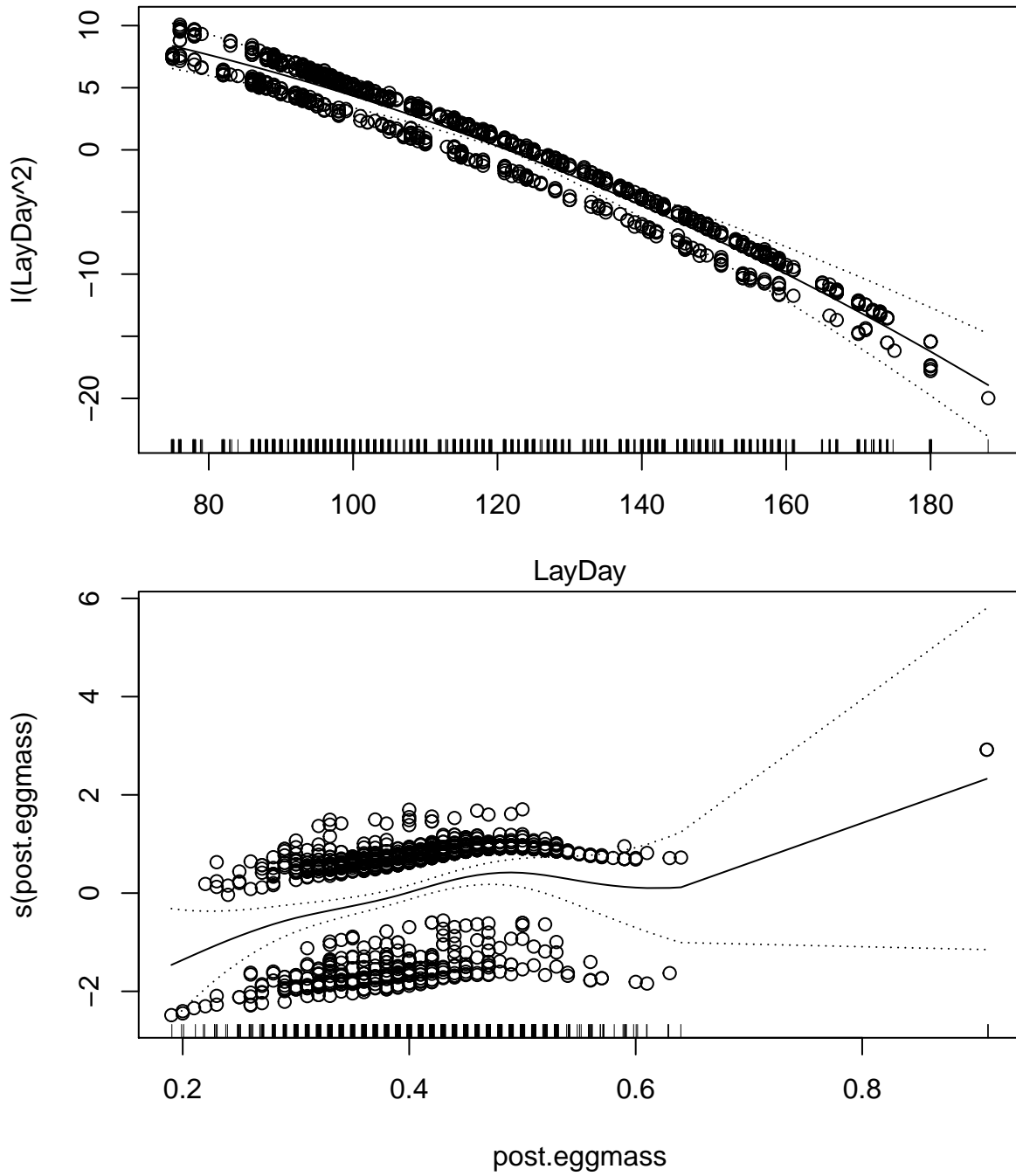
hatch.gam2<-gam(hatch~eggcode+s(LayDay)+I(LayDay^2)+s(post.eggmass), family=binomial, data=egg2, na.act.
anova (hatch.gam2)

## Anova for Nonparametric Effects
##           Npar Df Npar Chisq   P(Chi)
## (Intercept)
## eggcode
## s(LayDay)           3    24.4535 2.009e-05 ***
## I(LayDay^2)
## s(post.eggmass)     3     8.8104  0.03192 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

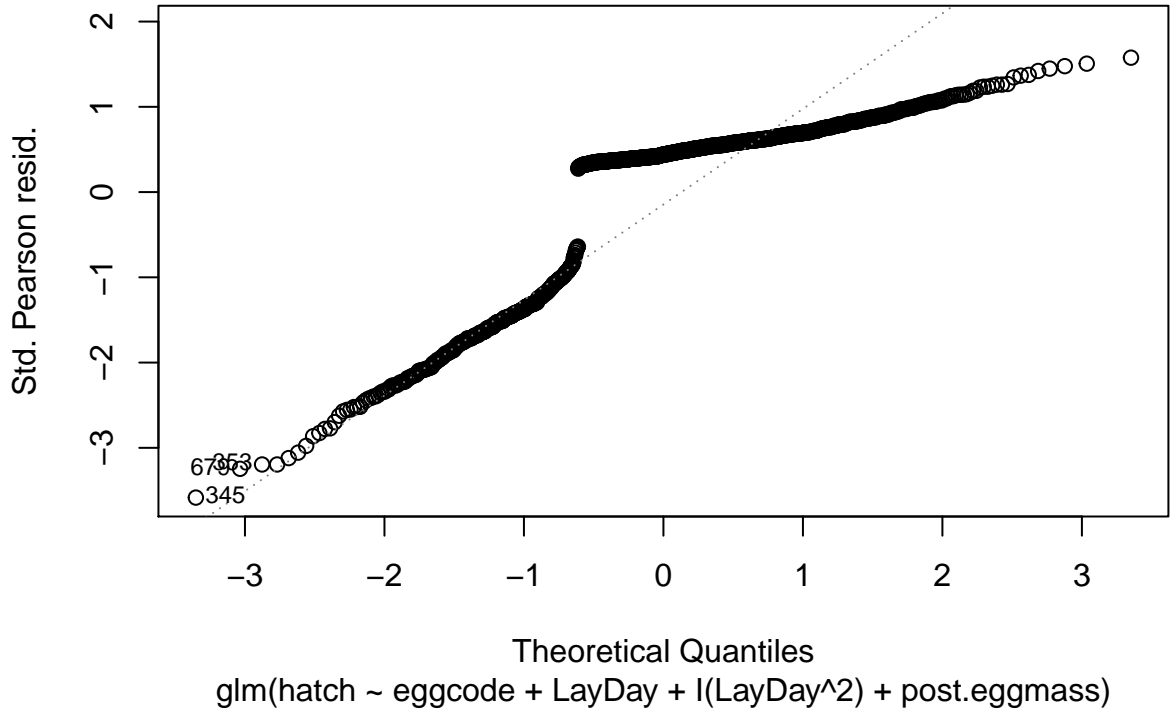
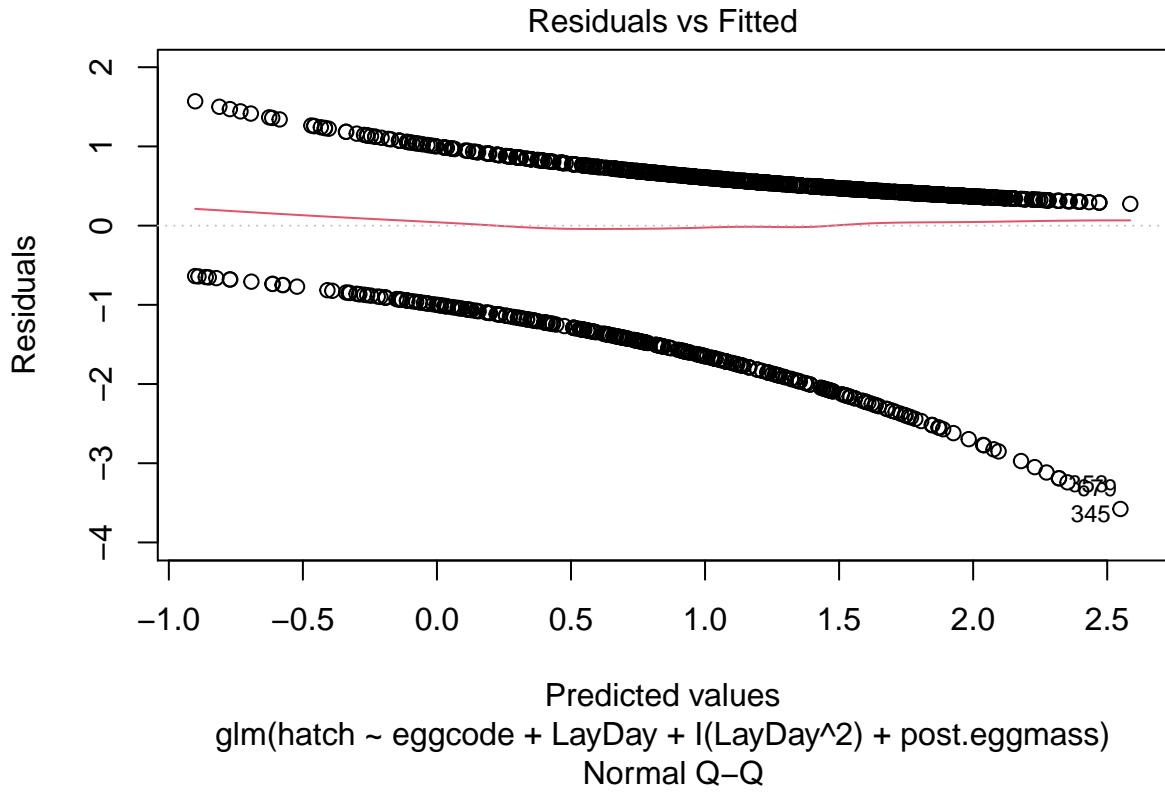
plot.Gam(hatch.gam2, se=T, residuals = T)

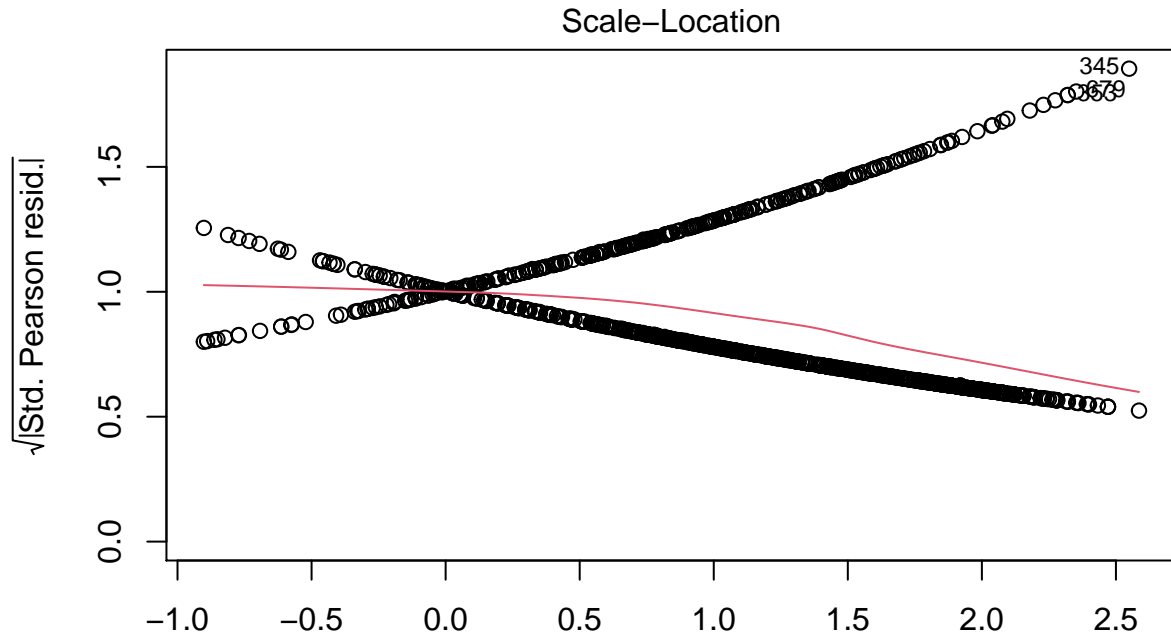
## Warning in gplot.default(x = c("M", "M", "M", "M", "M", "M", "M", "M", "M", "M", :
## The "x" component of "partial for eggcode" has class "character"; no gplot()
## methods available
```



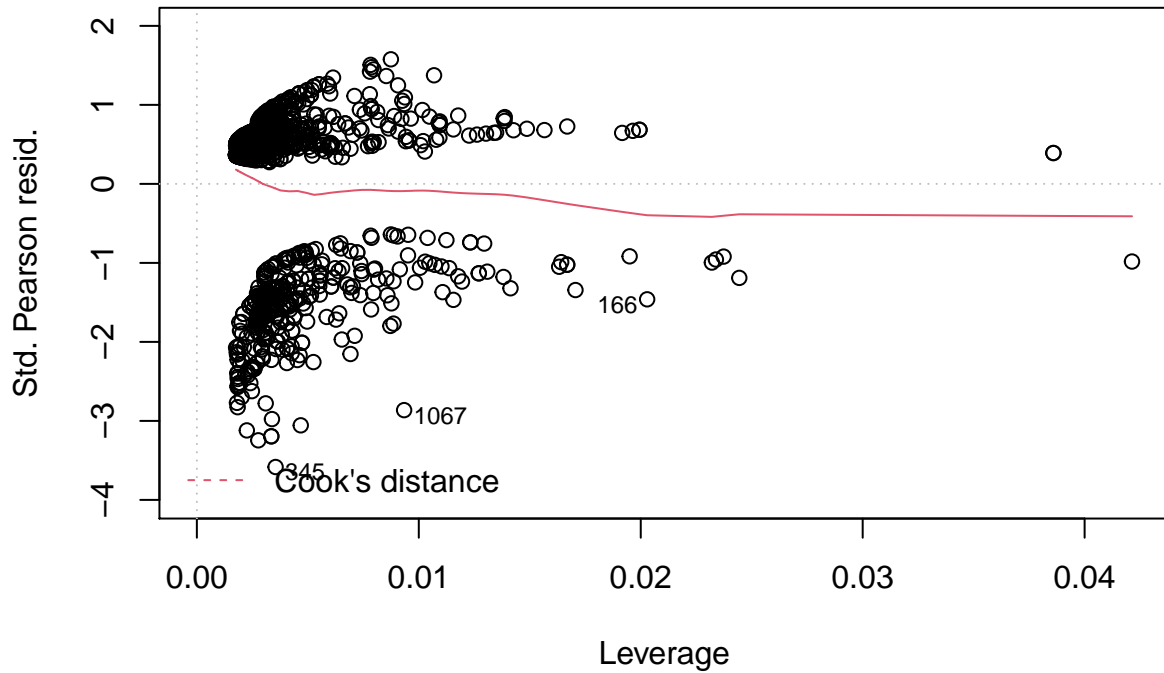


```
plot(hatch.glm2)
```





glm(hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass)
Residuals vs Leverage



glm(hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass)

It seems like the nonlinearity for posteggmass is driven by that one point. We will exclude it and rerun the model.

```
hatch.glm3<-glm(hatch~eggcode+LayDay+I(LayDay^2)+post.eggmass, family=binomial, data=egg2, na.action=na
```

We can compare the results of the two models

```
summary (hatch.glm2)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##      family = binomial, data = egg2, na.action = na.omit)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.2910  -1.0873   0.5929   0.7952   1.5763
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -9.982e+00  1.517e+00  -6.582 4.65e-11 ***
## eggcodeM     -5.332e-01  1.528e-01  -3.490 0.000483 ***
## LayDay        1.528e-01  2.458e-02   6.216 5.09e-10 ***
## I(LayDay^2)  -5.717e-04  9.905e-05  -5.772 7.85e-09 ***
## post.eggmass  3.945e+00  1.098e+00   3.591 0.000329 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1343.0  on 1242  degrees of freedom
## AIC: 1353
##
## Number of Fisher Scoring iterations: 4
```

```
summary (hatch.glm3)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + I(LayDay^2) + post.eggmass,
##      family = binomial, data = egg2, subset = -1067, na.action = na.omit)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3133  -1.0799   0.5908   0.7950   1.5780
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.014e+01  1.521e+00  -6.665 2.65e-11 ***
## eggcodeM     -5.073e-01  1.533e-01  -3.309 0.000936 ***
## LayDay        1.534e-01  2.463e-02   6.230 4.68e-10 ***
## I(LayDay^2)  -5.741e-04  9.924e-05  -5.785 7.27e-09 ***
## post.eggmass  4.227e+00  1.110e+00   3.809 0.000140 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1452.6  on 1245  degrees of freedom
## Residual deviance: 1338.5  on 1241  degrees of freedom
```

```
## AIC: 1348.5
##
## Number of Fisher Scoring iterations: 4
```

There is very little change in the parameters of the model by excluding that point.

Probit Model

Recall our original model:

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial)
```

In this case we are using the default link function for the binomial family. This is the “logit” function which converts probabilities into log odds = $\log(\text{prop}/(1-\text{prop}))$. An equivalent model would be:

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))
```

We could also specify

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(probit))
```

I am less familiar with the probit link but apparently these two links will often give similar results except when there are many very high or very low probabilities in which case probit is preferred.

```
hatch.glm<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))
summary (hatch.glm)
```

```
##
## Call:
## glm(formula = hatch ~ eggcode + LayDay + post.eggmass, family = binomial(logit),
##      data = egg2)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.3011  -1.1999   0.6390   0.8161   1.3262
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.847376   0.507975  -3.637 0.000276 ***
## eggcodeM     -0.509826   0.150804  -3.381 0.000723 ***
## LayDay        0.012742   0.002738   4.654 3.25e-06 ***
## post.eggmass  3.992286   1.105930   3.610 0.000306 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1375.4  on 1243  degrees of freedom
## AIC: 1383.4
##
## Number of Fisher Scoring iterations: 4
anova (hatch.glm, test="Chi")

## Analysis of Deviance Table
##
```

```
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                1246      1455.3
## eggcode             1   32.635      1245      1422.6 1.112e-08 ***
## LayDay              1   33.547      1244      1389.1 6.955e-09 ***
## post.eggmass       1   13.635      1243      1375.5 0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that this is the same approach as we used for an `lm`. The only difference is that now we are dealing with an F test. Note that the anova output assesses (in sequence) whether the addition of the next term significantly improves the fit of the model relative to the simpler model in which that term is not yet included. We can show the sequence of these tests as:

```
temp1<-glm(hatch~1, data=egg2, family=binomial(logit))
temp2<-glm(hatch~eggcode, data=egg2, family=binomial(logit))
temp3<-glm(hatch~eggcode+LayDay, data=egg2, family=binomial(logit))
temp4<-glm(hatch~eggcode+LayDay+post.eggmass, data=egg2, family=binomial(logit))

anova (hatch.glm, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: hatch
##
## Terms added sequentially (first to last)
##
##
##           Df Deviance Resid. Df Resid. Dev Pr(>Chi)
## NULL                1246      1455.3
## eggcode             1   32.635      1245      1422.6 1.112e-08 ***
## LayDay              1   33.547      1244      1389.1 6.955e-09 ***
## post.eggmass       1   13.635      1243      1375.5 0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova (temp1, temp2, temp3, temp4, test="Chi")
```

```
## Analysis of Deviance Table
##
## Model 1: hatch ~ 1
## Model 2: hatch ~ eggcode
## Model 3: hatch ~ eggcode + LayDay
## Model 4: hatch ~ eggcode + LayDay + post.eggmass
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      1246      1455.3
## 2      1245      1422.6  1   32.635 1.112e-08 ***
## 3      1244      1389.1  1   33.547 6.955e-09 ***
```



```
## 4      1243      1375.5  1   13.635  0.000222 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We wouldn't normally do this. Normally you would use the anova output to assess the significance of the terms in your model with the understanding that this is based on a Type I philosophy (i.e. order matters)

Note the reminder that terms are ordered from first to last.

Interpreting parameters in a GLiM

We will start simple

```
hatch.simple<-glm(hatch~post.eggmass, family=binomial, data=egg2)
summary(hatch.simple)
```

```
##
## Call:
## glm(formula = hatch ~ post.eggmass, family = binomial, data = egg2)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -2.279  -1.304   0.682   0.821   1.248
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -1.6059     0.3808  -4.217 2.48e-05 ***
## post.eggmass   6.5468     0.9623   6.803 1.02e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 1455.3  on 1246  degrees of freedom
## Residual deviance: 1405.1  on 1245  degrees of freedom
## AIC: 1409.1
##
## Number of Fisher Scoring iterations: 4
```

When you get a predicted value remember that it is on the log-odds scale because of the logit link function.

So if we wanted to calculate the predicted probability of survival for an egg that was 0.2g we would first get the predicted value on the log-odds scale.

```
-1.6059+6.5468*0.2
```

```
## [1] -0.29654
```

This is on the log-odds scale. We can then use a function in the boot package to back-transform this value. Note that the two colons just specifies that the function `inv.logit` is located within the boot package. This notation is not needed, but it can help clarify situations where there are two functions in different packages with the same name. It also helps to make it more clear to you that this is why I needed to load the boot package at the start of this document.

```
boot::inv.logit(-0.29654)
```

```
## [1] 0.4264035
```

We can check this against our calculation

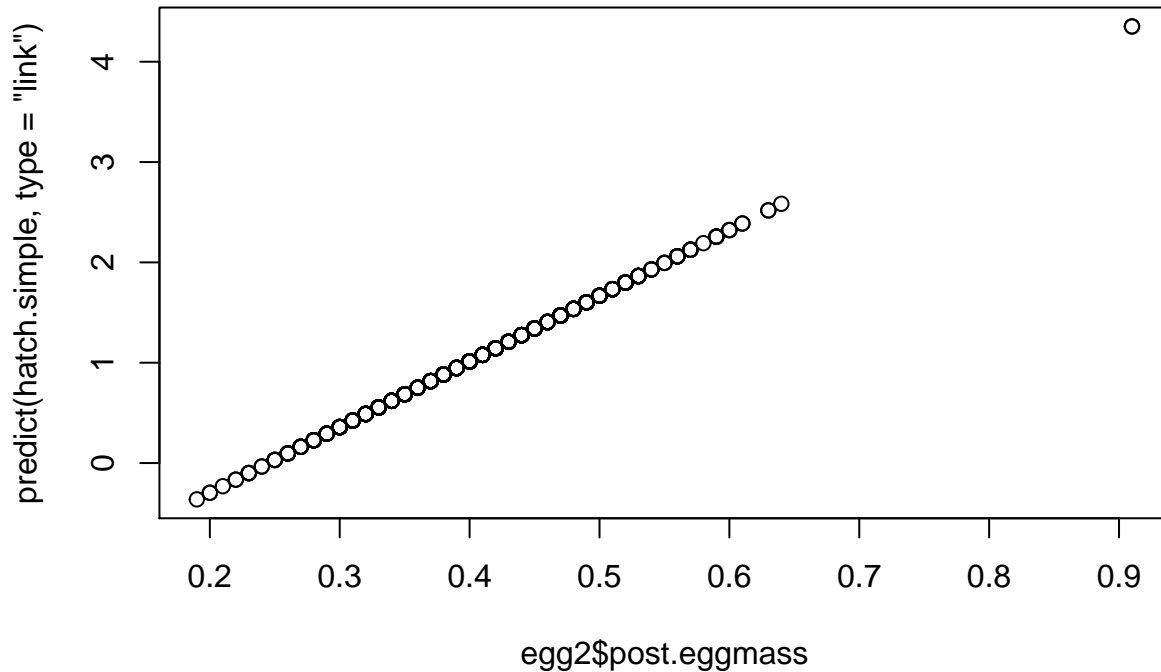
```
log(0.426/0.564)
```

```
## [1] -0.2806149
```

Therefore there is about a 43% chance that an egg that is 0.2g will survive until hatching.

We can also plot the predicted values against egg mass

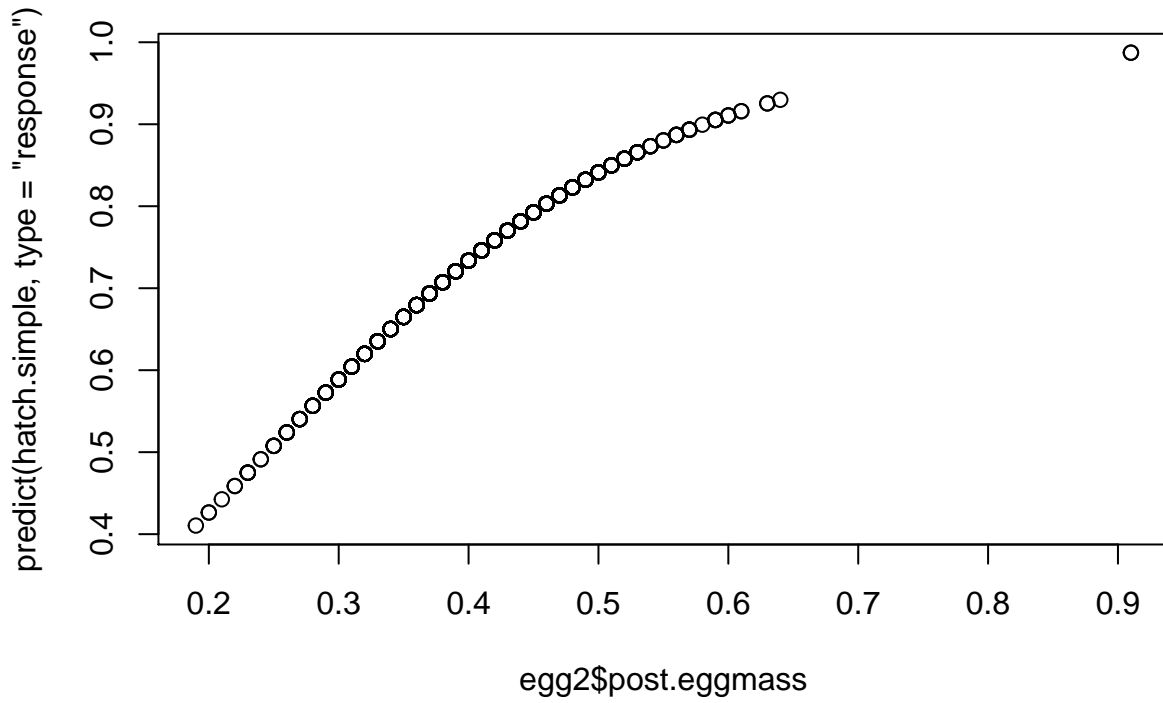
```
plot(egg2$post.eggmass, predict(hatch.simple, type="link"))
```



Note that this line has the slope and intercept that is presented in the model summary. However the logit link function means that these predicted values are not on the original scale that they were measured on.

We can instead get the predicted plot on the original raw scale by specifying type="response".

```
plot(egg2$post.eggmass, predict(hatch.simple, type="response"))
```

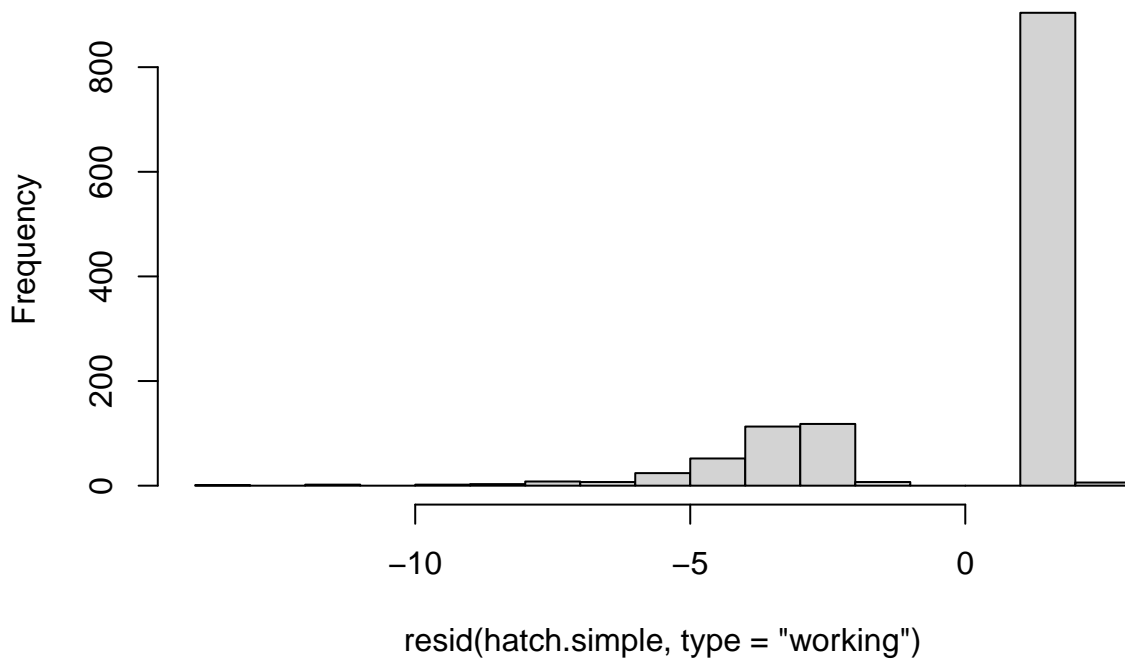


Residuals

With GLiM's there are also 4 different types of residuals

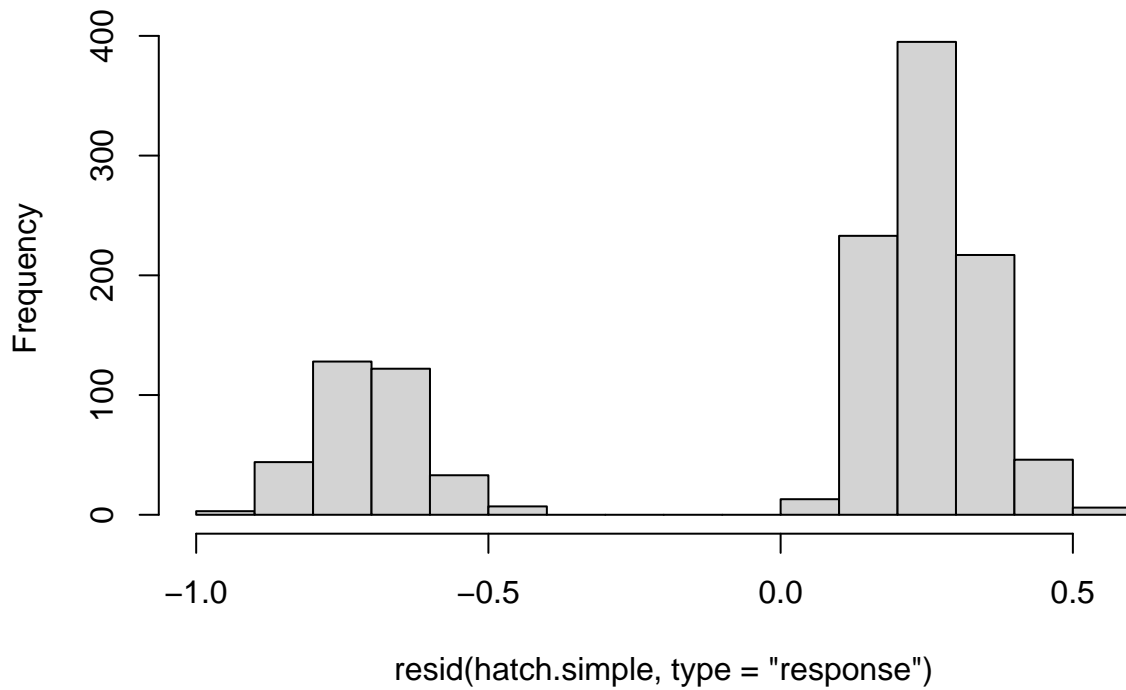
```
hist(resid(hatch.simple, type="working"))
```

Histogram of resid(hatch.simple, type = "working")



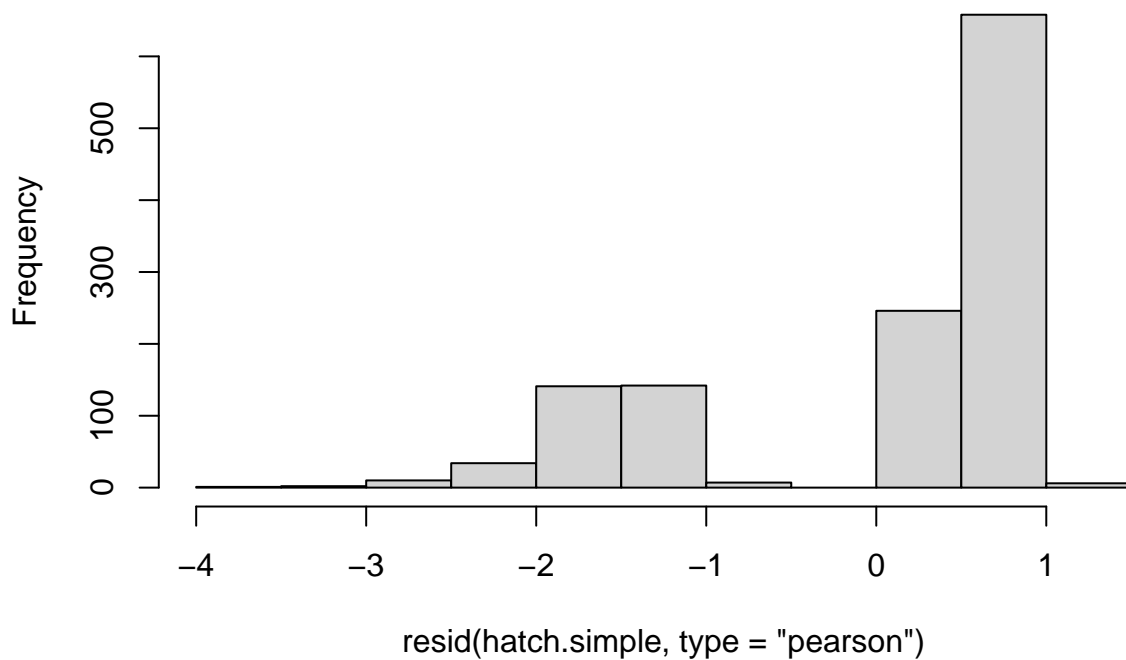
```
hist(resid(hatch.simple, type="response"))
```

Histogram of resid(hatch.simple, type = "response")



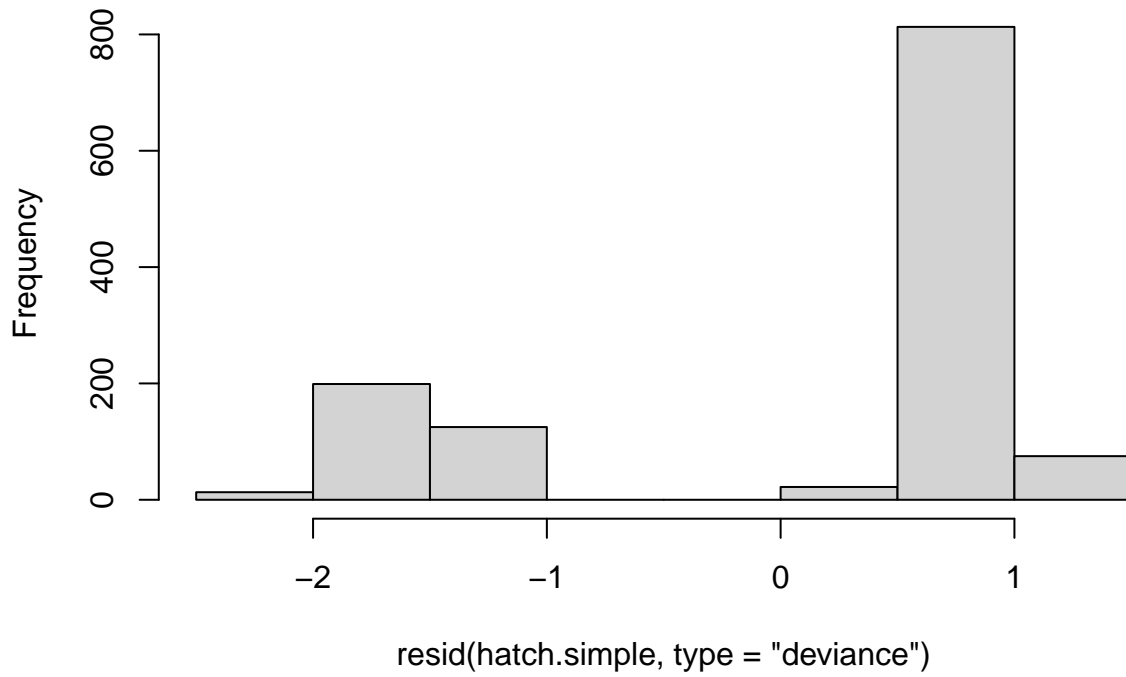
```
hist(resid(hatch.simple, type="pearson"))
```

Histogram of resid(hatch.simple, type = "pearson")



```
hist(resid(hatch.simple, type="deviance"))
```

Histogram of resid(hatch.simple, type = "deviance")



Deviance residuals are the best for diagnostic plots.